Unlink

Vulnerable to TOCTOU issues

Sean Barnum, Cigital, Inc. [vita[1]]

Copyright © 2007 Cigital, Inc.

2007-04-23

Part "Original Cigital Coding Rule in XML"

Mime-type: text/xml, size: 8585 bytes

Attack Category	• Path spoofing or confusion problem
Vulnerability Category	• Indeterminate File/Path • TOCTOU - Time of Check, Time of Use
Software Context	• File Management
Location	• unistd.h
Description	The unlink() function removes a link to a file. If path names a symbolic link, unlink() removes the symbolic link named by path and does not affect any file or directory named by the contents of the symbolic link. Otherwise, unlink() removes the link named by the pathname pointed to by path and decrements the link count of the file referenced by the link. The unlinkat() function also removes a link to a file. See fsattr(5). If the flag argument is 0, the behavior of unlinkat() is the same as unlink() except in the processing of its path argument. If path is absolute, unlinkat() behaves the same as unlink() and the dirfd argument is unused. If path is relative and dirfd has the value AT_FDCWD, defined in <fcntl.h>, unlinkat() also behaves the same as unlink(). Otherwise, path is resolved relative to the directory referenced by the dirfd argument. If the flag argument is set to the value AT_REMOVEDIR, defined in <fcntl.h>, unlinkat() behaves the same as rmdir(2) except in the processing of the path argument as described above. When the file's link count becomes 0 and no process has the file open, the space occupied by the file will be freed and the file is no longer accessible. If one or more processes have the file open when the last link is removed, the link is removed before unlink() or unlinkat() returns, but the removal of the file contents is postponed until all references to the file are closed.

1. http://buildsecurityin.us-cert.gov/bsi-rules/35-BSI.html (Barnum, Sean)

The path argument must not name a directory unless the process has appropriate privileges and the implementation supports using unlink() and unlinkat() on directories.

Upon successful completion, unlink() and unlinkat() will mark for update the st_ctime and st_mtime fields of the parent directory. If the file's link count is not 0, the st_ctime field of the file will be marked for update.

APIs

Function Name	Comments
_tunlink	use; win32
_unlink	use; win32
_wunlink	use; win32
unlink	use
unlinkat	

Method of Attack

The key issue with respect to TOCTOU vulnerabilities is that programs make assumptions about atomicity of actions. It is assumed that checking the state or identity of a targeted resource followed by an action on that resource is all one action. In reality, there is a period of time between the check and the use that allows either an attacker to intentionally or another interleaved process or thread to unintentionally change the state of the targeted resource and yield unexpected and undesired results.

A TOCTOU attack in regards to unlink() can occur when

a. A check for the existence of a file, for example, occurs.

b. An unlink command is executed.

Between a and b, an attacker could, for example, link the target file (the one to be unlinked) to a known file. The subsequent unlink would "unlink" the attacked file.

Exception Criteria

Solutions

Solution Applicability	Solution Description	Solution Efficacy
Generally applicable.	The most basic advice for TOCTOU vulnerabilities is to not perform a check before the use. This does not resolve the	Does not resolve the underlying vulnerability but limits the false sense of security given by the check.

		underlying issue of the execution of a function on a resource whose state and identity cannot be assured, but it does help to limit the false sense of security given by the check.	
	Generally applicable.	Limit the interleaving of operations on files from multiple processes.	Does not eliminate the underlying vulnerability but can help make it more difficult to exploit.
	Generally applicable.	Limit the spread of time (cycles) between the check and use of a resource.	Does not eliminate the underlying vulnerability but can help make it more difficult to exploit.
	Generally applicable.	Recheck the resource after the use call to verify that the action was taken appropriately.	Effective in some cases.

Signature Details	int unlink(const char *path); int unlinkat(int dirfd, const char *path, int flag);

Examples of Incorrect Code	`#include <unistd.h>`
	`char *path = "/modules/pass1";`
	`int unlink_status;`
	`struct stat stats;`
	`stat(path, &stats);`
	`...`
	`unlink_status = unlink(path);`

Examples of Corrected Code	`FILE *safe_open_wplus(char *fname)`
	`{`
	`struct stat lstat_info,`
	`fstat_info;`

```
FILE *fp; char *mode = "rb+"; /
*We perform our own truncation.*/
int fd;

if(lstat(fname, &lstat_info) == –
1) {
/* If the lstat() failed for
reasons other than the file
not existing, return 0,
specifying error. */
if( errno != ENOENT ) { return
0; }
if((fd = open(fname, O_CREAT|
O_EXCL| O_RDWR, 0600)) == –1) {
return 0; } mode = "wb";
}
else {
/* Open an existing file */
if((fd = open(fname, O_RDWR)) ==
–1) { return 0; }
if(fstat(fd, &fstat_info) == –1
||
lstat_info.st_mode !=
fstat_info.st_mode ||
lstat_info.st_ino !=
fstat_info.st_ino ||
lstat_info.st_dev !=
fstat_info.st_dev ) {

close(fd);
return 0;
}
/* Turn the file into an empty
file, to mimic w+ semantics. */
ftruncate(fd, 0);
}
/* Open a stdio file over the
low-level one */ fp = fdopen(fd,
mode); if(!fp) {
close(fd); unlink(fname); return
0;
}
return fp;
}
```

Source References

- Viega, John & McGraw, Gary. *Building Secure Software: How to Avoid Security Problems the Right Way*. Boston, MA: Addison-Wesley Professional, 2001, ISBN: 020172152X, ch 9
- man page for unlink()
- Microsoft Developer Network Library (MSDN)
- McGraw, Gary & Viega, John. "Building Secure Software: Race Conditions[2]." informit.com (2001).

Recommended Resource	
Discriminant Set	**Operating Systems**: • UNIX • Windows **Language**:

Cigital, Inc. Copyright

Copyright © Cigital, Inc. 2005-2007. Cigital retains copyrights to this material.

Permission to reproduce this document and to prepare derivative works from this document for internal use is granted, provided the copyright and “No Warranty” statements are included with all reproductions and derivative works.

For information regarding external or commercial use of copyrighted materials owned by Cigital, including information about “Fair Use,” contact Cigital at copyright@cigital.com[1].

The Build Security In (BSI) portal is sponsored by the U.S. Department of Homeland Security (DHS), National Cyber Security Division. The Software Engineering Institute (SEI) develops and operates BSI. DHS funding supports the publishing of all site content.

1. mailto:copyright@cigital.com